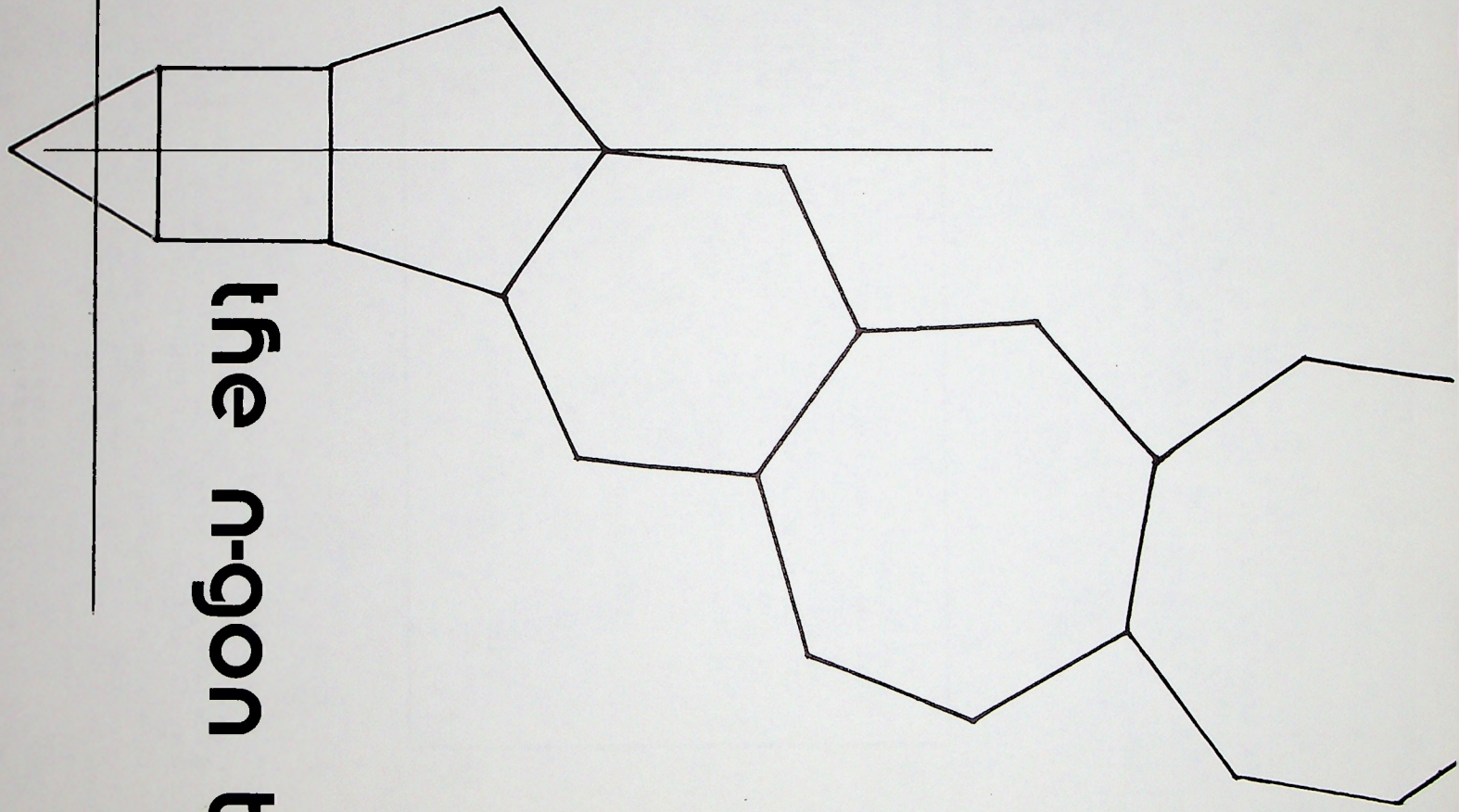
VOL 3 NO 1 JANUARY 1975



the n-gon trip

Popular Computing 22

PROBLEM **72**

Regular polygons of 3, 4, 5,...,97 sides, each with sides one unit long, are linked together as shown on the cover. The triangle has its center at the origin. For the polygons with an even number of sides, the direction of the chain is straight ahead. For those with an odd number of sides, the direction alternates right and left. Thus, after the 5, 9, 13,... sided polygons, the chain turns slightly to the right; for the 7, 11, 15,... sided polygons, it turns slightly to the left.

Problem: where will the center of the 97-gon be?

In issue No. 20, a dozen different algorithms were presented for calculating square root. David Ferguson (of Group/3) points out that one of the earliest machine algorithms should be added to the collection. The algorithm (of unknown authorship) dates back to the time when a divide operation on an automatic machine was a frill, and even if available was to be used as little as possible.

The Newton-Raphson scheme is applied to

$$y = 1/x^2 - N = 0$$

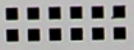
and results in the recursion

$$x_{n+1} = .5\, x_n(3 - Nx_n^2).$$

The method converges slowly, but has the virtue of requiring no divisions. When it converges, the required square root is given by Nx.

POPULAR COMPUTING is published monthly at Box 272, Calabasas, California 91302. Subscription rate in the United States is $18 per year, or $15 if remittance accompanies the order. For Canada and Mexico, add $4 per year to the above rates. For all other countries, add $6 per year to the above rates. Back issues $2 each. Copyright 1975 by POPULAR COMPUTING.

Publisher: Fred Gruenberger
Editor: Audrey Gruenberger
Associate Editors: David Babcock, Irwin Greenwald
Contributing editors: Richard Andree, Daniel D. McCracken, William C. McGee
Advertising manager: Ken W. Sims
Art Director: John G. Scott
Business Manager: Ben Moore



symposium 15

Every year since 1958, a one-day discussion session on computing has been held. At the 1973 session, held at the Airport Marina Hotel in Los Angeles, the attendees were:

Paul Armer, Center for Advanced Study in the Behavioral Sciences
Robert Bemer, Honeywell Information Systems
Erich Bloch, IBM
Fred Braddock, Informatics Inc.
Curtis Gerald, California Polytechnic State University, San Luis Obispo
George Glaser, AFIPS
Irwin Greenwald, Xerox Corporation
Fred Gruenberger, California State University, Northridge
Don Krehbiel, Santa Monica City College
Thomas R. Parkin, Control Data Corporation
Robert Reinstedt, The RAND Corporation

[A copy of the complete transcript of the symposium can be obtained for $10 from the Bureau of Business Services and Research, California State University, Northridge, 91324, the sponsor of the symposium.]

The 15th symposium had the topic "Exploring the Future." A modified Delphi technique was used to try to achieve a consensus on when certain milestones would be passed. For example, the attendees were polled in advance of the meeting for their opinion on the proposition "PL/I will be as dead as ALGOL is (in this country) in 1973; that is, no vendor will boast of offering PL/I as a language." On that particular item, the advance polling indicated a mean of 1988, with a low of 1973 and a high of after the year 2000. At the meeting on December 1, the discussion was aimed at accounting for the wide range of responses, in order to try to reach some agreement. The areas considered included the following:

1. The date when half the computing power of the U.S. would reside in what are now called mini computers.
2. The future of PL/I and APL.
3. The date when the world's chess champion would be a computer program.
4. The date when language translation, from idiomatic language A to idiomatic language B, would be economically feasible by machine.
5. The date when fingerprint recognition would be economically feasible by machine.
6. The date when computing would be a standard school subject in the same sense that algebra is today.
7. The date when more than half the states would require some system of licensing for computists.

As might be expected, no consensus was reached on most of the items. Some excerpts from the discussion are of interest. Bear in mind that the following quotations are taken somewhat out of context.

PARKIN: I look on computers as intelligence amplifiers; as drudgery-grinders; as tools in exactly the same sense as a lathe. Computers do precisely what we tell them to do. They will probably become as pervasive and all-encompassing in our lives as electric energy. I expect that computers will change civilization more than the industrial revolution did. I don't see how we can fault computers for the ills of our government. The 500-odd men in Congress worry about (1) getting reelected, (2) lining their pockets, and—maybe—(3) the country's problems. It's our fault if we don't set up the mechanisms for getting better people in government. You can't blame computers because people chose not to correlate data.

ARMER: One of the weaknesses of the Delphi technique is the difficulty of wording the questions so that they're unambiguous without at the same time revealing what the designer considers the "proper" answer. This is a fine example. The problem of fingerprint recognition has two distinct and widely differing meanings:

(1) Here is my fingerprint. Does it indeed match the one in your file labelled "Paul Armer"?
(2) Here is a fingerprint. Whose is it?

GRUENBERGER: I tried to word the item about PL/I very carefully. ALGOL is alive in Europe, but it is totally dead here. It's still available from many vendors, but they don't brag about it. You can't buy a machine on the basis of its ALGOL capability.

GREENWALD: My answer to that question (after the year 2000) assumed that there would be dialects of PL/I that would continue.

GRUENBERGER: Prof. Gerald and I were involved in the current procurement of new computers for the state college system. The committee we were on felt compelled to ask every college department (some 1200 of them) what programming languages they felt they would need in the 1975-1980 time period. There were some 800 responses, listing some 183 languages that someone considered essential to his work through 1980. The list included languages like SOAP and TYDAC, and five or six that no one (on the committee of 15 experts) could even identify. Now, you can't ask for bids on machines and require SOAP, since even IBM couldn't deliver that. The winners in the survey were—surprise—Fortran, COBOL, PL/I, and BASIC, and those are the only ones you can legally ask for anyway. In the same sense as the man who asks for SOAP, PL/I will surely be around in the year 2050 because there will be at least one clown who has to have it.

KREHBIEL: Will whole companies be using PL/I? Will 3-man service bureaus be using it? Will the University of California be using it?

BRADDOCK: There was a survey of some 900 IBM users in which 14% claimed to be using PL/I in some (unstated) way.

KREHBIEL: I'm under the impression that it takes a very large machine to run PL/I, and hence I conclude that only large corporations can use it.

PARKIN: But technology continues to improve. You ought to be able to implement PL/I on an 8K byte machine.

KREHBIEL: But right now it takes a big machine, doesn't it?

PARKIN: Yes, for the particular implementation that exists, but that's not the state of technology.

GREENWALD: The Burroughs 6700 has a design that should lend itself to an efficient PL/I compiler, both for compile time and run time. The point is that PL/I is attractive enough to be cast into hardware, and eventually the compile time will tend toward zero. The same architecture could be cast into smaller machines.

KREHBIEL: But that's some ways away from me. I'm a small user, and I don't rate a 370/167. I'm dealing with a Gremlin that has its tail end chopped off, and the operating system keeps feeling around for that missing piece. Give me PL/I in a 370/115 and I'll start being interested.

PARKIN: There's no real reason why PL/I couldn't be implemented on a mini computer before long.

KREHBIEL: I still don't understand what a mini is. I understand the characteristics of a computer, such as the fact that instructions and data are stored in the same medium and instructions can be treated as data by other instructions. But whenever I ask any vendor anything about "Can your machine do such and such?," the answer is always "yes." So what really differentiates the minis?

GREENWALD: I disagree that minis are going to take over the computing world. There's the question of centralization vs. decentralization. I think there's a big market for both sizes of machine, and I think that centralized computing will increase.

PARKIN: IBM will not discontinue the sale of big machines; there will always be a market for the biggest and most expensive machine. But more and more people are going to question the wisdom of having a super-large machine that is cut up, at great cost, into many little machines, which is what the users see. Technology will eventually produce small packages of computing power (defined any way you wish) accessible and available in clusters to the users. The *number* of minis will far exceed the number of other machines.

Let me try a provocative point. I run an advanced concepts research laboratory. One thing we worry about is the time when the hardware is so cheap that you could essentially give it away and charge only for the system or the software or something else. The cost per bit of storage or of logic element is ever-decreasing, and at a steady rate. It is easy to see ahead to the time when it will be feasible to produce something functionally equivalent to a 6600 in a package the size of a cigarette box, for which the most expensive part is the plug. How will we use the technology at that point? We continue to have dramatic breakthroughs in technology; they're evolutionary

but still dramatic. Such things drop the cost by an order of magnitude. Sometimes it takes a while before they are observable, but they do happen, and apparently without letup. How are we going to adapt to make use of those breakthroughs? It's this thinking that guided most of my responses. We have to look ahead to the time when bits, and logic elements, and redundancy will be so cheap as to be negligible. That's why I think, for example, that languages will proliferate, rather than die out.

GRUENBERGER: The current game among calculator users is "How many function buttons does your machine have?" Pretty soon it will be "How many words of addressable storage does your machine have?" And sometime after that it will be "How many program steps can your machine hold?" When a pocket machine has a button labelled "standard deviation," a lot of people are going to ask "What is that?" just as millions of people must now be observing that their machine has a button labelled "divide," and up to then they had never had any use for division, much less to 8 significant digits. If nothing else, these new machines are going to have a profound effect on understanding, by masses of people, of esoteric mathematical and scientific concepts.

GREENWALD: Will masses of people be able to deal with concepts like storage, sequencing, and complicated functions?

BEMER: Look what APL has done. The people who become familiar with APL think in terms of its functions, which are very powerful. They just naturally think at a much higher level. It may be that we can someday teach kids to start thinking at a higher level of abstraction.

GLASER: You're grossly underrating the customers. I know of many installations where the DP manager knows his business, and his management knows where the money goes. These men have stature, and common sense, and political clout. It's not universal, and it may never be; you can't stamp out idiocy. But I'm encouraged by what I see. The level of review committees is high, and by and large they're smart.

GREENWALD: Much of this management awareness and know-how was generated during the 1970 recession. Perhaps a 1974 recession will increase their awareness.

GLASER: I agree. People don't learn from an executive course or from a Fortran manual; managers learn when the Profit and Loss statement comes out. Along these lines, I'm a very strong advocate of charge back systems; I want the user to pay every nickel of the costs. There are exceptions, of course, but I know that with proper charge back, the quality of the work goes up and its reception is assured; everything gets better. It's painful, I know, since companies can say "This isn't our normal procedure; we don't charge for accounting services, for example." But accounting isn't discretionary, and DP systems should be, and when they're not, the chances of failure go 'way up.

ARMER: You're saying that you want feedback in a system.

GLASER: Yes, it's sharp pointed negative feedback, almost to the point of being punitive, but it has the right effect.

ARMER: If we assume constant productivity of systems programmers, and the demand increases, then what? Will the demand go up faster than productivity?

PARKIN: The cost of the hardware keeps going down. I predict that the demand for systems programming is going to go up, rapidly.

BEMER: The monetary feedback information will operate, when people observe that the systems people cost a fantastic amount relative to the hardware. To reduce those costs, people will turn to automated techniques for software.

PARKIN: Not in my lifetime.

ARMER: I wonder whether the hope for significant improvement in productivity isn't akin to the same hopes for machine translation, or machine chess.

BLOCH: No, it's a different kind of problem, and one that lends itself to new techniques. For example, we know how to apply engineering techniques to the production of software.

GREENWALD: But in the IBM studies, for example, it turns out that if you could double the amount of time actually spent on writing programs (versus everything else the programmer does), you'd still be under 2%.

BEMER: Let me put it this way. Programming is a tricky thought process. The tie-up comes (with long turnaround times) in getting back in context with those tricky thought processes. Just by shortening the turnaround time (to nearly zero), the programmer stays in context and productivity goes up.

GREENWALD: And all our tools have enabled us to go, in systems programming, from 30 checked out instructions per day down to 5 to 7 per day.

GRUENBERGER: In the scientific area, we have done certain problems once and for all; for example, the solution of simultaneous equations, or gear design, or Bessel function calculations. Isn't there a corresponding body of systems software problems that have been solved, so that each man doesn't have to solve them all over again? Doesn't the building block principle apply here, too?

BEMER: It's more difficult. You might like a packaged tax routine that could be plugged into any program that deals with taxes, but the tax laws are too varied to permit it.

GERALD: But couldn't we create tax modules, that could be parameterized and then collected to fit specific situations?

BRADDOCK: It depends, of course, on how you define systems software. We've all dealt with I/O instructions that deal directly with the peripheral devices. But today's systems programmers don't do that; they don't even know how tape or disk drives actually work, and they don't care. Their level of expertise is much different from that of systems programmers of ten years ago. A lot of people can turn out code in assembly language or Fortran or COBOL, but that doesn't make them systems programmers. We have developed a cadre of competent people who know their jobs, and they are developing the tools (or modules) that everyone else can use. One shouldn't generalize, but to my way of thinking, anyone who writes in Fortran is not a systems programmer; they are applications programmers getting a job done. We'll need a lot more of those.

BLOCH: I can't see what bearing the choice of language has on the matter. If he designs a system and uses Fortran, he's a systems programmer.

GREENWALD: Let's eliminate the semantic problem here. If he writes an operating system, or a language translator, he's a systems programmer and Braddock says there will be less such people. If he *uses* the product of a systems programmer, he's an applications programmer, and Braddock says there will be more such people.

PARKIN: I keep pointing out that the hardware is going to the point where we can give it away, and all we'll have left to sell will *be* systems.

GRUENBERGER: Tell me what I should tell my students (those who are headed toward careers in computing). Do I tell them that after 7 years or so they will be at peak salary unless they go into management?

GLASER: Yes, unless they pick up some merit badges along the way, such as knowledge of production control, or accounting systems, or manufacturing control, or go from sales to statistics to market research.

REINSTEDT: In other words, he must keep himself adaptable, and mobile, rather than narrow.

BRADDOCK: From management's point of view, a man should seek knowledge and constantly improve himself. The big trouble is that most people acquire only that knowledge that is essential to the project they've been assigned to. My big gripe is the man who is immersed in data base work (having been assigned to that task) who remains ignorant of another area (e.g., communications) which he should know about.

REINSTEDT: Here's another example. At one time, linear programming was a big thing. If we had five programmers whose specialty was linear programming, and they had learned nothing else, then they'd all be in trouble now, because linear programming just isn't in demand.

GREENWALD: We're being unfair. A person gets involved with a specific area, like linear programming, because that was the work he was assigned to. When a new problem in that area comes along, he gets it because he's the expert in it. And as long as he's involved with his specialty, we expect him to work at it, and we're not apt to encourage him to be studying other areas. I doubt that that will change.

BLOCH: That's true for a drill press operator, but a professional man has a responsibility to keep himself informed, at least, about other areas.

BEMER: Part of the problem is caused by the people themselves. The tenure in a particular assignment could be halved (say, three years writing Fortran compilers instead of six years at it) if they would learn to document what they had done so they could move on.

PARKIN: Fifteen years ago everyone in our field had a feeling of great excitement at being involved with this new high order of intellectual activity. Everyone could see years ahead of interesting new problems and applications, and everyone was learning at high speed. Today, that feeling seems to be gone. I am appalled at the 25 and 30-year-old people who have stopped learning; who say, in effect "I've learned the trade; I'm an expert; I don't need to learn anything else." They keep going at that level, and they're hacks. What apalls me is how the hack level is appearing at earlier and earlier ages. Maybe it's the "they aren't raising kids like they used to" syndrome.

GREENWALD: Those of us in this room all learned by experience, since that was the only way possible then. We all did everything. But today we can get in a young man who gets assigned to SYSGEN work, and pretty soon he's the local expert and can't be spared for anything else. He could quit and go somewhere else, but he can't get reassigned within his company; he's stuck. Even if he tries for reassignment, we always have deadlines to meet, and we seem to be better off letting him be stuck.

GRUENBERGER: I was startled by the responses to our question about the certification or licensing of programmers. Nearly everyone said "We aren't going to do that." I think we are, and that it may be forced on us in ways we won't like. Perhaps we could define the problem better by making an analogy to the mechanisms for the CPA. The rules for that—the avenues toward getting it—and the enforcement procedures—are all laid out, and they work, and they have been quite stable for over 25 years.

BLOCH: But the technology of the CPA has been the same for 300 years.

GRUENBERGER: Don't believe it. The accounting world changes pretty fast. The changes are not as fast as in our business (and they are far more orderly) but they are first order effects. For example, about 10 years ago they sent a CPA to jail, telling him "You should have known," and not accepting his plea that he didn't know of the shenanigans that were taking place in the firm he was auditing. For 25,000 CPA's in the country, the ball game changed its rules overnight. More recently, we've had Equity Funding, which will cause even more changes.

GLASER: Going back to computing, are the objections to certification and licensing due to a belief that we can't do it right, or that we shouldn't do it?

REINSTEDT: My position is that we can't possibly do it right (but that we're going to do it).

GLASER: If that's true, and it comes about anyway, what will happen? Will we find ourselfes with a lot of people who are certified but incompetent?

GRUENBERGER: Can we agree that the program has worked for the CPA's?

REINSTEDT: They are not all equal, but I get a distinct feeling of what constitutes a CPA, and I think most of us do. But try to extend that same notion to programmers.

BRADDOCK: An analogy with doctors may be appropriate. There is probably a written examination for them, but the real test is their apprenticeship, which goes on for several years. We will face the same problem, and our solution should probably be the same; namely, a long apprenticeship.

GRUENBERGER: I used the word "programmer" only in the catch phrase "Certified Public Programmer," but the question relates to certification of computer people in general. We should be asking, can a man be certified as knowledgable about computers and their uses?

ARMER: For whom will such people work? Would they work for firms that send a man in to certify another firm's programs? In other words, would they function the way CPA's do?

GREENWALD: Companies hire accountants and they hire programmers. They can get a certified accountant if they wish, or they can also get one who is not certified. They could do the same thing with programmers.

GLASER: The CPA certificate has motivated a lot of people to try to reach a stated level of knowledge. It has done a lot for the accounting profession. True, a man crams to pass that set of exams, but it's unfair to conclude that he then stops learning.

REINSTEDT: I'm all for motivating people to learn more and upgrade themselves. But when you take the tests and get the certificate, what are you then certified to do?

GLASER: Well, it's much like requiring a Boy Scout to take a 50 mile hike. It won't guarantee his ability to survive in the woods, but it's evidence of some level of capability, and several such requirements put him ahead of the boy who hasn't done them. As things stand now, you have no evidence at all from anyone who walks in the door and says "I'm a programmer."

ARMER: I require that the guy I hire has a college degree. It's not that the degree has given him anything specific, but simply that the probability of finding a good man in that population is much higher than that of finding a good man in the non-degree population. The degree is a sifting device, and the certificate could serve the same purpose.

GLASER: The Harvard Law School graduate may not be better trained that the graduate of Podunk, but statistically he's a better bet. If nothing else, his survival ability is better.

REINSTEDT: But don't tell me he's certified.

GLASER: Not as an individual. But in hiring him, your risk is lower if that's all you know.

REINSTEDT: Then the term "certified" is a misnomer; worse, it's a non sequitur.

KREHBIEL: It is any worse than what we expect from a man who can call himself a lawyer?

REINSTEDT: When I go to a lawyer, I know what I can expect from him.

PARKIN: You do? You must be as ignorant about law as most of us are about medicine, then.

REINSTEDT: But what's the alternative? Given a legal problem, I must go to a lawyer, and I know what to expect from him.

GREENWALD: Isn't all this just a substitute for a programming aptitude test? Those were designed to save personnel departments some time and effort.

PARKIN: They turn out to be only IQ tests.

REINSTEDT: Not "turn out to be"; they were *taken* from IQ tests.

GLASER: I think the present DPMA tests are better, for their numbers, than any of us would acknowledge. Clearly, those tests do not apply to numerical analysts, or scientific programmers, or the artificial intelligence boys; the tests just don't apply.

REINSTEDT: In analyzing the results of the last DPMA exams, they broke out those who were taking the test for the first time. Those who had majored in data processing in college came in second from the last (next to accountants) and under education majors, math majors, engineers, and everyone else, on the first two parts of the test. For the other parts, they were on the bottom.

GLASER: Sure; they learned DP from numerical analysts and mathematicians. They didn't learn from people who had practical experience in the DP world.

GRUENBERGER: All this is charming, but totally irrelevant to the question, which was When will half the states require some sort of certificate?—good, bad, or indifferent. You guys are all busy designing the perfect certificate, which isn't the point. It seems to me that if we have two more Equity Funding scandals within six months of each other, then about two months later more than half the states will require licensing of computer people, and they won't care how good it is.

PARKIN: A lot of doctors have killed their patients, but that is not the mechanism that led to the medical examining boards we now have. The medical profession decided to police itself, and quietly keep its mistakes from the public view.

GRUENBERGER: That only supports my statement. We ought to keep our mistakes to ourselves, too, and act to do it before it's forced on us.

GREENWALD: Us old people might have to protect ourselves from the young people.

KREHBIEL: Then you go on to restrict entry into the field, and you add grandfather clauses (in our case, literally).

GLASER: We joke about it, but in five years or so, the economic pressure on the 45-year-olds will be strong enough to make that more likely to happen than not.

REINSTEDT: I guess the answer to the question is that we will have certified programmers pretty soon, and it will be meaningless.

GRUENBERGER: The question was *when*?

BRADDOCK: I voted for a late year, when it might mean something.

GREENWALD: I would now vote much earlier. The politicians will say "We recognize the problem, and we have done something about it."

KREHBIEL: If it takes them as long to recognize this problem as it did the oil shortage, we have a lot of time.

GLASER: The people in DPMA who run the certification program know that it isn't as good as it ought to be; that it needs fixing; and that they acknowledge that it needs fixing. Few people would defend it as the ultimate.

BEMER: My motto in computing has always been these five words: *Do something small useful now.*

REINSTEDT: The certification boys are about to do something large useless now.

Log 22	1.342422680822206235963938865967517268474892071928562
Ln 22	3.091042453358315853479175699423305867897206988297672
$\sqrt{22}$	4.690415759823429554565630113544466280588228353411737
$\sqrt[3]{22}$	2.802039330655387120665677385665894017585798218769268
$\sqrt[5]{22}$	1.855600736258084334732770521321674794599490537151919
$\sqrt[7]{22}$	1.555158536763463318507348720266186388028707231984833
$\sqrt[10]{22}$	1.362204366553743041822749217702391289113218086364251
$\sqrt[100]{22}$	1.031393112229484533249772315571564116212442930070430
e^{22}	3584912846.131591561681159945978420689222693065037274 934931971841555786919360975329001975957054
π^{22}	86556004191.98134152251135804670373660115625743426539 53850048107098781181657147912973613134003
$\tan^{-1} 22$	1.525373047373319604208237289205537102419159594135981
22^{100}	17469001504088245598835440079017000089716288685957915 23527044718998741634271162410507673817696314640559387 34482112142486751362076901376

The figures on this page are the eleven possible ways in which four pentagons can be joined at their edges.

PC22-10

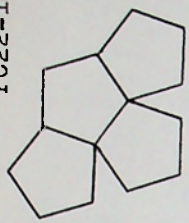
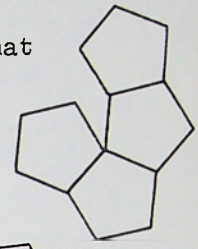
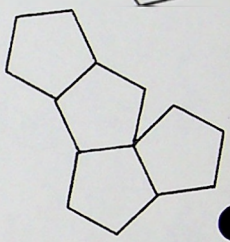


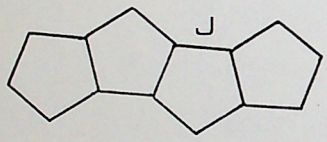
Table H shows the present state of knowledge about such polyominoes made up of squares, triangles, and hexagons. The table is furnished by Thomas R. Parkin, of Control Data Corporation, who first calculated the values for squares, up to case 15. The values for 16, 17, and 18 were calculated by Prof. W. Fred Lunnon for his PhD thesis.

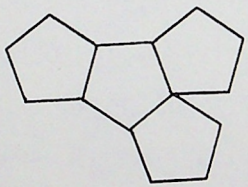
Work on polyominoes has been done only for squares, triangles, and hexagons, because those are the polygons that can tile the plane. Pentagons are a different animal.

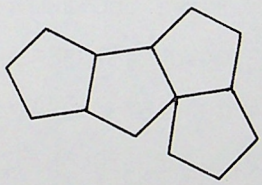


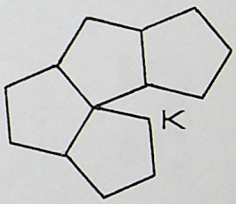
To find out how many pentagonal polyominoes there are for case 5, one could follow this straightforward algorithm: append a pentagon to every possible side of the shapes on this page, and then eliminate the duplicates from the resulting set of figures. Both parts of that algorithm might be difficult to apply. For example, there are clearly 14 places where the next pentagon can be appended to Figure J, but it is not immediately clear how many can be appended to Figure K. As Mr. Parkin points out, "Unfortunately, it requires trigonometry to know if a pentagon can be added to some figures in particular places. Thus, the growth of figures as N increases becomes a question of how accurately one can compute distances and, since trigonometric functions are transcendental, there is no precise integer answer."

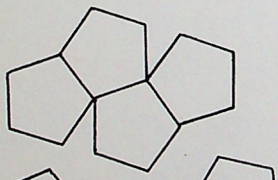


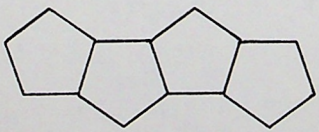


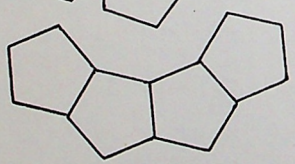






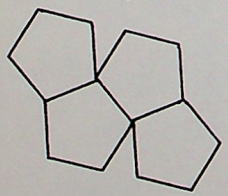






Note: the counts in Table H are for the free shapes, as noted, but the polyominoes on this page are of fixed shapes; that is, left and right versions of the same shape are both shown.

Using either fixed or free shapes, the Problem is to extend Table H in the column for pentagons.



Pentagonal Polyominoes

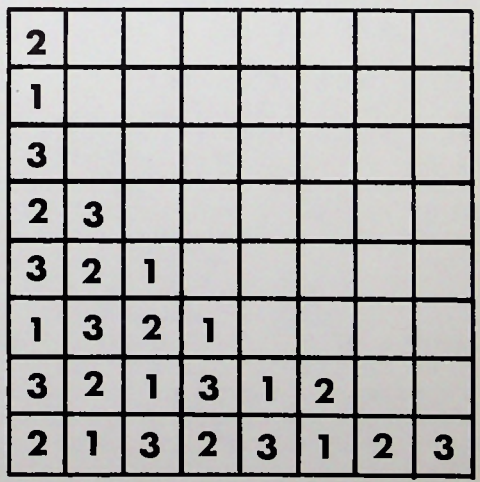
Table of known information on polyominoes

N	Squares	Triangles	Hexagons	(Pentagons)
1	1	1	1	1
2	1	1	1	1
3	2	1	3	2
4	5	3	7	7
5	12	4	22	
6	35	12	82	
7	108	24	333	
8	369	66	1448	
9	1285	160	6572	
10	4655	448	30,490	
11	17,073	1186	143,552	
12	63,600	3334	683,101	
13	238,591	9235		
14	901,971	26,166		
15	3,426,576	73,983		
16	13,079,255	211,297		
17	50,107,911			
18	192,622,052			

H

Note: These counts are for the free shapes; i.e., those which are free to rotate and reflect in the plane.

The two-dimensional Flagstone Problem



The border pattern shows a solution to the Flagstone Problem:

> A man has flagstones of three different colors. How can he lay them so that no pattern of colors is immediately repeated; that is, so that no consecutive pair of stones has the same color; no consecutive pairs have the same colors in the same order; no three stones show the same sequence of colors as the preceding three; and so on for any sets of N stones.

(The shapes in the border design have no meaning; the three colors are represented by white, black, and centered dot.)

The Flagstone Problem is number F13 in the book Problems for Computer Solution (Gruenberger and Jaffray). It is Problem 33 of the fifth book of Problematical Recreations, Litton Industries. It first appeared in "Unending Chess, Symbolic Dynamics and Problems in Semigroups," Marston Morse and Gustav Hedlund, in the Duke Mathematical Journal, Vol. 11, March, 1944. In mathematical terms, as the Flagstone Problem, it appeared in the problem section of the American Mathematical Monthly, submitted by Hugh Noland, June-July 1963. A solution by C. H. Braunholtz appeared in the same issue.

The problem makes an interesting exercise in computer logic. Consider this sequence:

1232132312321231323121312321323123212313231 21

and the logic of extending it. The next digit cannot be a 1. It also cannot be a 2, since that would repeat the 2-digit sequence 1212. But it also cannot be a 3, since that would repeat the 23-digit sequence starting at the beginning. Therefore, it is necessary to back off and change the last given digit from 1 to 3, and then proceed forward again. It may be necessary to back off many digits.

Braunholtz showed a method of constructing such sequences of any length, so the Flagstone Problem remains only as an exercise in computer coding. The task at hand is to extend the problem to two dimensions: in the array shown, each row, reading from left to right, and each column, reading from bottom to top, conforms to the one-dimensional case. Can the pattern be extended indefinitely?

Speaking of Languages

ROBERT TEAGUE

In the last issue (PC21-12) we discussed some of the workings of the PLATO system for CAI and also its unique terminal. This month, I would like to turn our attention to TUTOR, the language that makes PLATO work.

Modes of operation. There are three modes of operation within the system: (1) system mode; (2) author mode; and (3) student mode. Each of these modes is available to a user only when he has the right access code in his user number. They are downward inclusive; that is, someone in system mode can work in author mode or student mode if he desires, but not vice versa.

The system mode allows a programmer to make changes to the PLATO system itself. This is possible because PLATO is written in TUTOR just as any instructor-prepared course material would be. (One of the old criteria of a good language was whether or not the language compiler could be written in the language. Here the answer is, yes it can and is.)

Author mode is the necessary mode of operation for an instructor (or any user) to be in to create new materials for the system. In this mode he can create, edit, and execute the courses he is developing for the system.

Student mode can only be used to run existing course materials. However, student users normally have priority to use of the system and can frequently get on when an author cannot.

Program structure. The "program" (the term isn't used) in TUTOR is called a course. Every student or author must be listed as being enrolled in each of the courses on the system in order to access those materials. The course is broken down in two ways: (1) in a physical breakdown into blocks, or (2) in a logical breakdown into lessons. A block is given a name by which changes to it are made, and may contain one or more lessons. The lesson consists of the materials to be presented upon a subject. A lesson could, for example, be written on a topic in chemistry, or accounting, or any other discipline. Within each lesson are one or more units. Units comprise the materials to be presented on the display screen at one time, and since they are named, are the logical transfer points in the lesson.

The language commands. The TUTOR language commands have two parts: the command and the tag. The command gives the operation to be performed, while the tag gives various information depending on the command. For example, a "write" (all TUTOR commands are given in lower case) command would have a tag giving the information to be displayed on the screen, while a "jump" would have the name of the unit to which the transfer will be made. Although not exhaustive, the accompanying table gives all the commands needed to write complete course materials in any field.

The lesson has two states of operation that must be considered before going into the actual commands. Normal state will simply execute the display, computation, or utility commands in sequence. Once the "arrow" command is used to elicit a response from the student, however, the state will be shifted to judging state. In this state, a judging command must be used before the three other types of commands will have effect again (this is called "satisfying the arrow"). For example, in the code

```
arrow   1510
answer  4
write   Correct.  Very good.
answer  5
write   Close enough, but it's really 4.
wrong   6
write   Try again.
```

if the input for the "arrow" is 4, the first write will be executed and the next four lines of code will be skipped. If the answer were 5, the second write is executed and the next two lines skipped. But if the input were 6, "Try again." would be displayed and the system would automatically (because the "wrong" command would judge the response 6 as incorrect) go back to the "arrow" command to elicit another response. Because of these states and the way they operate, TUTOR is not actually a totally sequential language. The list of "answer" and "wrong" commands acts like a jump vector, with the one matching the input being selected to execute next. This situation occurs in a few other places as well.

Next month we will go into the commands with sample lessons written in TUTOR, to give more feel for the capabilities of the language.

Table of TUTOR Commands

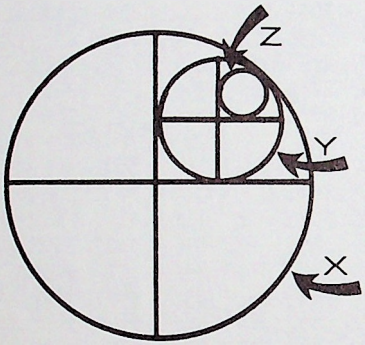
Display	Judging	Computation	Utility
at	arrow	calc	unit
write	answer	randu	next
erase	wrong	define	jump
draw	no	addl	pause
circle	ok	subl	do
show			
ansv			
wrongv			
size			
long			

Nested Circles

PROBLEM **75**

In a circle of unit radius, another circle is drawn in one quadrant, tangent to the quadrant lines and the original circle. In this new circle, the process is repeated; that is, the inner circle is quartered and a new circle is drawn in one of the quarters as before.

The original circle (X in the figure) has an area of pi square units. What is the total area occupied by the infinite sequence of smaller circles? (Circle Y has an area of about .54 square units; circle Z has an area of about .09 square units; thus the total area occupied by just those two circles is about .63 square units.) The summing process should begin with circle Y.



Check Protection

PROBLEM **76**

In the Check Writing Problem (Problem L2 in Problems for Computer Solution, Gruenberger and Jaffray, Wiley, 1965), amounts up to $499.99 for a check are to be translated into words, as for example:

FOUR HUNDRED NINETY NINE AND 99/100.

Considering only whole dollar amounts, starting with 1, what is the first appearance of each amount that requires more space on the check? If a check protection symbol (**) is printed just to the left of the dollar amount, for what amounts will that symbol first move further to the left? In other words, extend this list:

**ONE
**THREE
**ELEVEN
**THIRTEEN
**SEVENTEEN
**TWENTY ONE
**TWENTY THREE
**SEVENTY THREE
**ONE HUNDRED ONE
**ONE HUNDRED THREE
**ONE HUNDRED ELEVEN

PC22-16

Printed by Argold Press Inc.
16921 Ventura Boulevard, Encino

THE FIRST 1000 ODD PRIME NUMBERS

Those underlined mark the first appearance of each larger difference between successive primes. For example, the first appearance of a difference of 18 occurs between 523 and 541.

3	5	7	11	13	17	19	23	29	31	37	41	43	47	53	59
61	67	71	73	79	83	89	97	101	103	107	109	113	127	131	137
139	149	151	157	163	167	173	179	181	191	193	197	199	211	223	227
229	233	239	241	251	257	263	269	271	277	281	283	293	307	311	313
317	331	337	347	349	353	359	367	373	379	383	389	397	401	409	419
421	431	433	439	443	449	457	461	463	467	479	487	491	499	503	509
521	523	541	547	557	563	569	571	577	587	593	599	601	607	613	617
619	631	641	643	647	653	659	661	673	677	683	691	701	709	719	727
733	739	743	751	757	761	769	773	787	797	809	811	821	823	827	829
839	853	857	859	863	877	881	883	887	907	911	919	929	937	941	947
953	967	971	977	983	991	997	1009	1013	1019	1021	1031	1033	1039	1049	1051
1061	1063	1069	1087	1091	1093	1097	1103	1109	1117	1123	1129	1151	1153	1163	1171
1181	1187	1193	1201	1213	1217	1223	1229	1231	1237	1249	1259	1277	1279	1283	1289
1291	1297	1301	1303	1307	1319	1321	1327	1361	1367	1373	1381	1399	1409	1423	1427
1429	1433	1439	1447	1451	1453	1459	1471	1481	1483	1487	1489	1493	1499	1511	1523
1531	1543	1549	1553	1559	1567	1571	1579	1583	1597	1601	1607	1609	1613	1619	1621
1627	1637	1657	1663	1667	1669	1693	1697	1699	1709	1721	1723	1733	1741	1747	1753
1759	1777	1783	1787	1789	1801	1811	1823	1831	1847	1861	1867	1871	1873	1877	1879
1889	1901	1907	1913	1931	1933	1949	1951	1973	1979	1987	1993	1997	1999	2003	2011
2017	2027	2029	2039	2053	2063	2069	2081	2083	2087	2089	2099	2111	2113	2129	2131
2137	2141	2143	2153	2161	2179	2203	2207	2213	2221	2237	2239	2243	2251	2267	2269
2273	2281	2287	2293	2297	2309	2311	2333	2339	2341	2347	2351	2357	2371	2377	2381
2383	2389	2393	2399	2411	2417	2423	2437	2441	2447	2459	2467	2473	2477	2503	2521
2531	2539	2543	2549	2551	2557	2579	2591	2593	2609	2617	2621	2633	2647	2657	2659
2663	2671	2677	2683	2687	2689	2693	2699	2707	2711	2713	2719	2729	2731	2741	2749
2753	2767	2777	2789	2791	2797	2801	2803	2819	2833	2837	2843	2851	2857	2861	2879
2887	2897	2903	2909	2917	2927	2939	2953	2957	2963	2969	2971	2999	3001	3011	3019
3023	3037	3041	3049	3061	3067	3079	3083	3089	3109	3119	3121	3137	3163	3167	3169
3181	3187	3191	3203	3209	3217	3221	3229	3251	3253	3257	3259	3271	3299	3301	3307
3313	3319	3323	3329	3331	3343	3347	3359	3361	3371	3373	3389	3391	3407	3413	3433
3449	3457	3461	3463	3467	3469	3491	3499	3511	3517	3527	3529	3533	3539	3541	3547
3557	3559	3571	3581	3583	3593	3607	3613	3617	3623	3631	3637	3643	3659	3671	3673
3677	3691	3697	3701	3709	3719	3727	3733	3739	3761	3767	3769	3779	3793	3797	3803
3821	3823	3833	3847	3851	3853	3863	3877	3881	3889	3907	3911	3917	3919	3923	3929
3931	3943	3947	3967	3989	4001	4003	4007	4013	4019	4021	4027	4049	4051	4057	4073
4079	4091	4093	4099	4111	4127	4129	4133	4139	4153	4157	4159	4177	4201	4211	4217
4219	4229	4231	4241	4243	4253	4259	4261	4271	4273	4283	4289	4297	4327	4337	4339
4349	4357	4363	4373	4391	4397	4409	4421	4423	4441	4447	4451	4457	4463	4481	4483
4493	4507	4513	4517	4519	4523	4547	4549	4561	4567	4583	4591	4597	4603	4621	4637
4639	4643	4649	4651	4657	4663	4673	4679	4691	4703	4721	4723	4729	4733	4751	4759
4783	4787	4789	4793	4799	4801	4813	4817	4831	4861	4871	4877	4889	4903	4909	4919
4931	4933	4937	4943	4951	4957	4967	4969	4973	4987	4993	4999	5003	5009	5011	5021
5023	5039	5051	5059	5077	5081	5087	5099	5101	5107	5113	5119	5147	5153	5167	5171
5179	5189	5197	5209	5227	5231	5233	5237	5261	5273	5279	5281	5297	5303	5309	5323
5333	5347	5351	5381	5387	5393	5399	5407	5413	5417	5419	5431	5437	5441	5443	5449
5471	5477	5479	5483	5501	5503	5507	5519	5521	5527	5531	5557	5563	5569	5573	5581
5591	5623	5639	5641	5647	5651	5653	5657	5659	5669	5683	5689	5693	5701	5711	5717
5737	5741	5743	5749	5779	5783	5791	5801	5807	5813	5821	5827	5839	5843	5849	5851
5857	5861	5867	5869	5879	5881	5897	5903	5923	5927	5939	5953	5981	5987	6007	6011
6029	6037	6043	6047	6053	6067	6073	6079	6089	6091	6101	6113	6121	6131	6133	6143
6151	6163	6173	6197	6199	6203	6211	6217	6221	6229	6247	6257	6263	6269	6271	6277
6287	6299	6301	6311	6317	6323	6329	6337	6343	6353	6359	6361	6367	6373	6379	6389
6397	6421	6427	6449	6451	6469	6473	6481	6491	6521	6529	6547	6551	6553	6563	6569
6571	6577	6581	6599	6607	6619	6637	6653	6659	6661	6673	6679	6689	6691	6701	6703
6709	6719	6733	6737	6761	6763	6779	6781	6791	6793	6803	6823	6827	6829	6833	6841
6857	6863	6869	6871	6883	6899	6907	6911	6917	6947	6949	6959	6961	6967	6971	6977
6983	6991	6997	7001	7013	7019	7027	7039	7043	7057	7069	7079	7103	7109	7121	7127
7129	7151	7159	7177	7187	7193	7207	7211	7213	7219	7229	7237	7243	7247	7253	7283
7297	7307	7309	7321	7331	7333	7349	7351	7369	7393	7411	7417	7433	7451	7457	7459
7477	7481	7487	7489	7499	7507	7517	7523	7529	7537	7541	7547	7549	7559	7561	7573
7577	7583	7589	7591	7603	7607	7621	7639	7643	7649	7669	7673	7681	7687	7691	7699
7703	7717	7723	7727	7741	7753	7757	7759	7789	7793	7817	7823	7829	7841	7853	7867
7873	7877	7879	7883	7901	7907	7919	7927								